(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104834995 A
(43)申请公布日 2015.08.12

(21)申请号 201510188745.X

(22)申请日 2015.04.20

(71)申请人 安徽师范大学
地址 241002 安徽省芜湖市弋江区九华南路189号科技服务部

(72)发明人 张佩云 凤麒

(74)专利代理机构 北京润平知识产权代理有限公司 11283
代理人 董彬

(51)Int.Cl.
G06Q 10/06(2012.01)

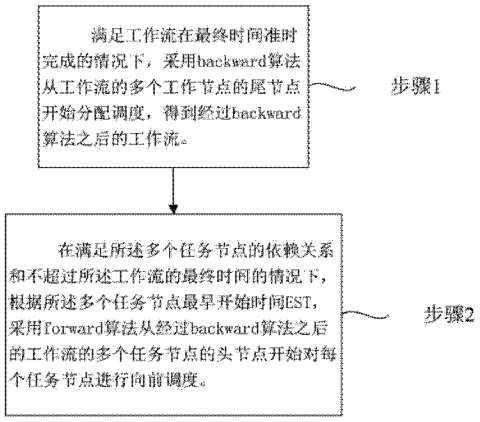
权利要求书2页 说明书13页 附图4页

(54)发明名称

基于云计算的工作流双向调度方法

(57)摘要

本发明公开了一种基于云计算的工作流双向调度方法，该工作流双向调度方法包括：步骤1，满足工作流在最终时间准时完成的情况下，采用backward算法从工作流的多个任务节点的尾节点开始分配调度，得到经过backward算法之后的工作流；步骤2，在满足所述多个任务节点的依赖关系和不超过所述工作流的最终时间的情况下，根据所述多个任务节点最早开始时间EST，采用forward算法从经过backward算法之后的工作流的多个任务节点的头节点开始对每个任务节点进行向前调度。该基于云计算的工作流双向调度方法节约了用户成本，满足用户灵活调度的个性化需求。



CN 104834995 A

1. 一种基于云计算的工作流双向调度方法，其特征在于，该工作流双向调度方法包括：

步骤 1，满足工作流在最终时间准时完成的情况下，采用 backward 算法从工作流的多个任务节点 v_i 的尾节点开始分配调度，得到经过 backward 算法之后的工作流；

步骤 2，在满足所述多个任务节点 v_i 的依赖关系和不超过所述工作流的最终时间的情况下，根据所述多个任务节点 v_i 最早开始时间 EST，采用 forward 算法从经过 backward 算法之后的工作流的多个任务节点 v_i 的头节点开始对每个任务节点 v_i 进行向前调度。

2. 根据权利要求 1 所述的基于云计算的工作流双向调度方法，其特征在于，所述步骤 1 包括：

步骤 S101，计算得到所有任务节点 v_i 的多个工作区间，根据所述工作区间的最迟结束时间 LFT 将多个所述工作区间从大到小进行排序，从所述任务节点 v_i 的尾节点开始依次执行步骤 S102；

步骤 S102，当所述任务节点 v_i 的所有直接后继节点所分配的虚拟机的类型相同于所述任务节点 v_i 所映射的虚拟机的类型，且该虚拟机在所述任务节点 v_i 的工作区间内空闲，则将所述任务节点 v_i 分配到该虚拟机上，否则，执行步骤 S103；

步骤 S103，求出任务节点 v_i 与所述任务节点 v_i 的直接后继节点的传输时间，比较所述任务节点 v_i 的多个直接后继节点的最迟结束时间和所述传输时间之和，得出所述和的最小值所对应的直接后继节点；

步骤 S104，当所述和的最小值所对应的直接后继节点所分配的虚拟机的类型与所述任务节点 v_i 所映射的虚拟机的类型相同，且该虚拟机在所述任务节点 v_i 的工作区间内空闲，则将所述任务节点 v_i 分配到该虚拟机上，否则，执行步骤 S105；

步骤 S105，当所述任务节点 v_i 所在类型的已经分配的虚拟机在所述任务节点 v_i 的工作区间内空闲，则将所述任务节点 v_i 分配到该虚拟机中，否则，执行步骤 S106；

步骤 S106，重新租用所述任务节点 v_i 所在类型的虚拟机，则将所述任务节点 v_i 分配到所述虚拟机中。

3. 根据权利要求 2 所述的基于云计算的工作流双向调度方法，其特征在于，所述工作区间从所述任务节点 v_i 的最迟开始时间 LST 至传输到该任务节点 v_i 的直接后继节点的完成时间之间的区间。

4. 根据权利要求 3 所述的基于云计算的工作流双向调度方法，其特征在于，在所述步骤 S102 和步骤 S103 中，当所述直接后继节点为所述任务节点 v_i 的唯一直接后继节点，且所述直接后继节点所分配的虚拟机的类型相同于所述任务节点 v_i 所映射的虚拟机的类型，则所述工作区间省去相邻节点的传输完成时间。

5. 根据权利要求 1 所述的基于云计算的工作流双向调度方法，其特征在于，所述步骤 2 包括，

步骤 S201，将经过 backward 算法之后的工作流中的头节点放入集合 SET 中；

步骤 S202，分别计算集合 SET 中的任务节点 v_i 在步骤 1 中所分配到的虚拟机以及与所述虚拟机同类的虚拟机的多个空闲时间戳；

步骤 S203，根据所述多个空闲时间戳计算得到每个空闲时间所对应的以下因素：最早开始时间影响因素 ESTF、浪费时间影响因素 WTF 和额外租用的费用影响因素 ACF；

步骤 S204，通过如下公式得到任务最优放置指数：

Index = ESTF×a+WTF×b+ACF×c，

其中，a、b、c 是权重，且 a、b、c 的范围都为 [0,1]；

步骤 S205，比较多个所述空闲时间戳所对应的任务最优放置指数的大小得到最小的最优放置指数，将集合 SET 中的任务节点 v_i 放入所述最小的最优放置指数所对应的空闲时间戳得到 FeasibleSet 集合；

步骤 S206，当所述任务节点 v_i 有直接后继节点时，将集合 SET 中的任务节点 v_i 替换成所述任务节点 v_i 的直接后继节点，否则，返回执行步骤 S202。

6. 根据权利要求 5 所述的基于云计算的工作流双向调度方法，其特征在于，在所述步骤 S203 中，计算最早开始时间影响因素 ESTF 的方法包括：

最早开始时间影响因素 ESTF 等于实际开始时间 AST 与最早开始时间之差 LST 除以最迟开始时间 EST。

7. 根据权利要求 5 所述的基于云计算的工作流双向调度方法，其特征在于，在所述步骤 S203 中，计算浪费时间影响因素 WTF 的方法包括：

浪费时间影响因素 WTF 等于在预设周期的时间内左边的空闲时间戳与右边的空闲时间戳的和除以所述预设周期的时间。

8. 根据权利要求 5 所述的基于云计算的工作流双向调度方法，其特征在于，在所述步骤 S203 中，计算额外租用的费用影响因素 ACF 的方法包括：

额外租用的费用影响因素 ACF 等于增加的额外虚拟机的预设周期数乘以预设周期除以所述任务节点 v_i 的工作区间。

9. 根据权利要求 1-8 中任意一项所述的基于云计算的工作流双向调度方法，其特征在于，该工作流双向调度方法还包括步骤 3，

根据每个虚拟机的费用 price(t) 通过如下公式计算已经分配的虚拟机的总费用 $Cost_{sum}$：

$$Cost_{sum}=\sum_{t=1}^{n}\left|d_{vi}^{t}/60\right|\times price\ (t)\ ,$$

其中，price(t) 代表第 t 种类型虚拟机每小时的收费标准，n 表示有 n 种虚拟机；d_{vi}^{t} 表示任务节点 v_i 在所分配的 t 类虚拟机上执行所需要的时间且 $d_{vi}^{t}/60$ 向上取整。

10. 根据权利要求 1-8 中任意一项所述的基于云计算的工作流双向调度方法，其特征在于，该工作流双向调度方法还包括：将分配调度后的任务节点 v_i 进行标记。

基于云计算的工作流双向调度方法

技术领域

[0001] 本发明涉及云计算的领域，具体地，涉及一种基于云计算的工作流双向调度方法。

背景技术

[0002] 工作流调度是一种多约束满足问题，例如对于工作流中的任务之间相互依赖关系、任务与任务之间的传输时间、任务类型的约束满足。云资源提供商往往会基于租赁的销售模式以及基于使用量和性能标准的计费模式对用户应用所提供的计算服务进行收费。不同的调度算法直接关系到云计算的整体稳定性，并且在保证整个工作流按时完成的情况下，尽可能为用户节省成本。从云服务供应商角度出发，既要保证云端服务质量(QoS)，又要考虑到负载均衡问题，以及利益最大化问题。因此，需要设计良好的调度算法，以实现云计算资源更好地共享。另外，在调度过程中，往往需要考虑到用户偏好问题。

[0003] 工作流任务调度是一个 NP-hard 问题，目前大多数调度算法从云服务商角度出发，最大化地使用云端的固定资源数量，从而产生最大效益，面临着把工作流中的任务理想化成同一种类型任务、没有考虑到云端虚拟机按小时收费的策略、没有把云中资源想象成同构资源、没有考虑到预算的限制以及未考虑到任务之间的通讯量等问题，容易造成在实际执行过程中用户成本的增加。此外，现有技术认为云资源是单一化的，而这与现实是不相符，因为在云端其资源总量、类别数量都比较多，不可能单一化；另外也有采用 deadline-assignment 策略，将整个工作流的最终时间 (deadline) 约束到各个子任务中，存在着资源利用率不高的问题；现有技术还研究了关于云计算的资源动态分配与共享技术，主要集中在无用户差别情况下的资源动态分配策略，但云计算最终对象是不同的用户，而用户不同的行为习惯决定了不同的服务质量要求。此外，由于云计算环境下按小时收费的策略，但是用户面临着如何减少因剩余时间(partial hour)统计不合理而额外花费的问题。例如，用户的某任务总共需要调度执行 1.2 小时，但云端服务提供商要按 2 小时收取费用，额外的 0.8 小时的收费给用户造成了不少损失，如果云端避免这 0.8 小时的使用，既可以避免用户的额外损失，又可以实现云端服务提供商的自身利益的最大化。因此，需要设计合理的调度算法。

发明内容

[0004] 本发明的目的是提供一种基于云计算的工作流双向调度方法，该基于云计算的工作流双向调度方法克服了现有技术中的虚拟机资源利用效率不高、没有考虑到用户差别化的问题，节约了用户成本，满足用户灵活调度的个性化需求。

[0005] 为了实现上述目的，本发明提供了一种基于云计算的工作流双向调度方法，该工作流双向调度方法包括：

[0006] 步骤 1，满足工作流在最终时间准时完成的情况下，采用 backward 算法从工作流的多个任务节点 v_i 的尾节点开始分配调度，得到经过 backward 算法之后的工作流；

[0007] 步骤 2，在满足所述多个任务节点 v_i 的依赖关系和不超过所述工作流的最终时间

的情况下，根据所述多个任务节点 v_i 最早开始时间 EST，采用 forward 算法从经过 backward 算法之后的工作流的多个任务节点 v_i 的头节点开始对每个任务节点 v_i 进行向前调度。

[0008] 优选地，所述步骤 1 包括：

[0009] 步骤 S101，计算得到所有任务节点 v_i 的多个工作区间，根据所述工作区间的最迟结束时间 LFT 将多个所述工作区间从大到小进行排序，从所述任务节点 v_i 的尾节点开始依次执行步骤 S102；

[0010] 步骤 S102，当所述任务节点 v_i 的所有直接后继节点所分配的虚拟机的类型相同于所述任务节点 v_i 所映射的虚拟机的类型，且该虚拟机在所述任务节点 v_i 的工作区间内空闲，则将所述任务节点 v_i 分配到该虚拟机上，否则，执行步骤 S103；

[0011] 步骤 S103，求出任务节点 v_i 与所述任务节点 v_i 的直接后继节点的传输时间，比较所述任务节点 v_i 的多个直接后继节点的最迟结束时间和所述传输时间之和，得出所述和的最小值所对应的直接后继节点；

[0012] 步骤 S104，当所述和的最小值所对应的直接后继节点所分配的虚拟机的类型与所述任务节点 v_i 所映射的虚拟机的类型相同，且该虚拟机在所述任务节点 v_i 的工作区间内空闲，则将所述任务节点 v_i 分配到该虚拟机上，否则，执行步骤 S105；

[0013] 步骤 S105，当所述任务节点 v_i 所在类型的已经分配的虚拟机在所述任务节点 v_i 的工作区间内空闲，则将所述任务节点 v_i 分配到该虚拟机中，否则，执行步骤 S106；

[0014] 步骤 S106，重新租用所述任务节点 v_i 所在类型的虚拟机，则将所述任务节点 v_i 分配到所述虚拟机中。

[0015] 优选地，所述工作区间从所述任务节点 v_i 的最迟开始时间 LST 至传输到该任务节点 v_i 的直接后继节点的完成时间之间的区间。

[0016] 优选地，在所述步骤 S102 和步骤 S103 中，当所述直接后继节点为所述任务节点 v_i 的唯一直接后继节点，且所述直接后继节点所分配的虚拟机的类型相同于所述任务节点 v_i 所映射的虚拟机的类型，则所述工作区间省去相邻节点的传输完成时间。

[0017] 优选地，所述步骤 2 包括，

[0018] 步骤 S201，将经过 backward 算法之后的工作流中的头节点放入集合 SET 中；

[0019] 步骤 S202，分别计算集合 SET 中的任务节点 v_i 在步骤 1 中所分配到的虚拟机以及与所述虚拟机同类的虚拟机的多个空闲时间戳；

[0020] 步骤 S203，根据所述多个空闲时间戳计算得到每个空闲时间所对应的以下因素：最早开始时间影响因素 ESTF、浪费时间影响因素 WTF 和额外租用的费用影响因素 ACF；

[0021] 步骤 S204，通过如下公式得到任务最优放置指数：

[0022] $Index = ESTF \times a + WTF \times b + ACF \times c$，

[0023] 其中，a、b、c 是权重，且 a、b、c 的范围都为 [0,1]；

[0024] 步骤 S205，比较多个所述空闲时间戳所对应的任务最优放置指数的大小得到最小的最优放置指数，将集合 SET 中的任务节点 v_i 放入所述最小的最优放置指数所对应的空闲时间戳得到 FeasibleSet 集合；

[0025] 步骤 S206，当所述任务节点 v_i 有直接后继节点时，将集合 SET 中的任务节点 v_i 替换成所述任务节点 v_i 的直接后继节点，否则，返回执行步骤 S202。

[0026] 优选地，在所述步骤 S203 中，计算最早开始时间影响因素 ESTF 的方法包括：

[0027] 最早开始时间影响因素 ESTF 等于实际开始时间 AST 与最早开始时间之差 LST 除以最迟开始时间 EST。

[0028] 优选地，在所述步骤 S203 中，计算浪费时间影响因素 WTF 的方法包括：

[0029] 浪费时间影响因素 WTF 等于在预设周期的时间内左边的空闲时间戳与右边的空闲时间戳的和除以所述预设周期的时间。

[0030] 优选地，在所述步骤 S203 中，计算额外租用的费用影响因素 ACF 的方法包括：

[0031] 额外租用的费用影响因素 ACF 等于增加的额外虚拟机的预设周期数乘以预设周期除以所述任务节点 v_i 的工作区间。

[0032] 优选地，该工作流双向调度方法还包括步骤 3，

[0033] 根据每个虚拟机的费用 price(t) 通过如下公式计算已经分配的虚拟机的总费用 $Cost_{sum}$：

[0034] $$Cost_{sum}=\sum_{t=1}^{n}\left|d_{vi}^{t}/60\right|\times price\ (t)$$，

[0035] 其中，price(t) 代表第 t 种类型虚拟机每小时的收费标准，n 表示有 n 种虚拟机；d_{vi}^{t} 表示任务节点 v_i 在所分配的 t 类虚拟机上执行所需要的时间且 $d_{vi}^{t}/60$ 向上取整。

[0036] 优选地，该工作流双向调度方法还包括：将分配调度后的任务节点 v_i 进行标记。

[0037] 通过上述的实施方式，本发明的基于云计算的工作流双向调度方法针对当前研究中存在的因剩余时间而导致的虚拟机资源利用效率不高、没有考虑到用户差别化等问题，从用户角度出发，针对虚拟机按小时收费模型，提出一种动态的双向调度算法，即从尾节点到头节点反向进行一次排序，再正向进行一次排序，在满足用户指定的截止时间的同时，又可以根据剩余时间实际情况动态地对虚拟机租用数量进行调节，从而进一步节约用户成本，满足用户灵活调度的个性化需求。

[0038] 本发明的其他特征和优点将在随后的具体实施方式部分予以详细说明。

附图说明

[0039] 附图是用来提供对本发明的进一步理解，并且构成说明书的一部分，与下面的具体实施方式一起用于解释本发明，但并不构成对本发明的限制。在附图中：

[0040] 图 1 是说明本发明的优选实施方式的工作流实例示意图；

[0041] 图 2 是说明本发明的优选实施方式的 backward 算法调度最终结果示意图；

[0042] 图 3 是说明本发明的优选实施方式的浪费时间影响因素 WTF 实例的示意图；

[0043] 图 4 是说明本发明的优选实施方式的额外租用的费用影响因素实例的示意图；

[0044] 图 5 是说明本发明的优选实施方式的 v_5 分配之前的虚拟机状态的实例的示意图；

[0045] 图 6 是说明本发明的优选实施方式的排除 c 之后 v_5 在虚拟机上两种可选择状态的示意图；

[0046] 图 7 是说明本发明的优选实施方式的虚拟机 v_5 分配后最终状态的示意图；

[0047] 图 8 是说明本发明的最优选方式的虚拟机调度整个工作流的最终结果的示意图；

[0048] 图 9 是说明本发明的 θ 取 0.15 时本文算法在执行费用与其它算法对比的效果图；

[0049] 图 10 是说明本发明的 θ 取 0.3 时本文算法在执行费用与其它算法对比的效果

图；以及

[0050] 图 11 是说明本发明的一种基于云计算的工作流双向调度方法的流程图。

具体实施方式

[0051] 以下结合附图对本发明的具体实施方式进行详细说明。应当理解的是，此处所描述的具体实施方式仅用于说明和解释本发明，并不用于限制本发明。

[0052] 在本发明的一种基于云计算的工作流双向调度方法的具体实施方式中，如图 11 所示，该工作流双向调度方法包括：

[0053] 步骤 1，满足工作流在最终时间准时完成的情况下，采用 backward 算法从工作流的多个任务节点 v_i 的尾节点开始分配调度，得到经过 backward 算法之后的工作流；

[0054] 步骤 2，在满足所述多个任务节点 v_i 的依赖关系和不超过所述工作流的最终时间的情况下，根据所述多个任务节点 v_i 最早开始时间 EST，采用 forward 算法从经过 backward 算法之后的工作流的多个任务节点 v_i 的头节点开始对每个任务节点 v_i 进行向前调度。

[0055] 通过上述的实施方式，本发明的首先采用 backward 算法从尾节点开始调度，在工作流的 deadline 时刻准时完成，同时保证每个任务都正好在最迟时刻完成。其次采用 forward 算法，根据虚拟机的空闲情况，动态调节任务的时间戳，根据每个任务的最早开始时间，从头节点开始，对每个任务进行向前调度，该任务标记为已分配任务，之后再动态调整后续任务的最早开始时间，在调度时，对同类型虚拟机上的任务，在不违反任务之间相互依赖关系以及不超出工作流 deadline 时，尽可能地合并在一个收费周期（预设为 60 分钟），从而不需要重新租用额外的虚拟机时间，以尽可能地为用户节省费用。

[0056] 在考虑到了尽可能减少任务之间的传输时间的同时，为用户制定了三种选择模式（在定义 7 详细介绍），用户可以根据实际需要进行选择调度模式。

[0057] 以下结合附图 1 对本发明进行进一步的说明，图 1 是具体的一种工作流实例示意图，

[0058] 1、图 1 所示的工作流总共有 7 个节点，v_1 到 v_7 代表各个任务，箭头方向从父任务节点指向子任务节点，表示调度时节点之间的先后关系，箭头上的数字代表两个节点之间的通信时长（单位：分钟）。

[0059] 令 $W = \{v_1, \cdots, v_n\}$ 表示一个有 n 个任务（节点）的工作流，W 中除首、尾任务外，其它任务都有直接父任务和直接子任务。用集合 $E = \{e_{ij}\}$ 表示父子任务之间拥有的直接有向边，其中，$e_{ij} = (v_i, v_j)$，$(1 <= i < j <= n)$。只有当某子任务所有父任务都执行完成后，该子任务（直接后继节点）才可以开始执行。当子任务与其父任务在同一个虚拟机上执行时，数据传输时间忽略不计，用 $TT(v_i, v_j)$ 来表示任务 v_i 和任务 v_j 之间的传输时间，用 D 表示工作流 W 的截止时间 deadline。

[0060] 与双向调度算法有关的定义如下：

[0061] 定义 1，工作流最迟结束时间 LFT(latest finish time)：$LFT(v_{exit}) = D$，

[0062] $$LFT(v_c) = \min LFT(v_i) - TT(v_i, v_c) - d^t_{vi}。$$

[0063] 在定义 1 中，v_{exit} 表示工作流的尾节点（该节点只是一个虚拟的最后节点，与其它节点之间并无通信量，本身也无计算量），v_i 和 v_c 均为工作流中的节点，且 v_c 是 v_i 的直接

后继节点，d_{vi}^{t}代表任务 v_i在所分配的虚拟机 t 上的执行时间。

[0064] 定义 2，最早开始时间 EST(earliest start time)：EST(ventry) = 0，EST

[0065] $(v_i) = \max EST(v_p) + TT(v_p, v_i) + d_{vp}^{t}$。

[0066] 在定义 2 中，ventry 表示工作流的首节点，v_p和 v_i均为工作流中的节点且 v_p是 v_i的父节点，d_{vp}^{t}代表任务 v_p在所分配的虚拟机 t 上的执行时间。

[0067] 定义 3，最迟开始时间 LST(latest start time)：$LST(v_i) = LFT(v_i) - d_{vi}^{t}$。

d_{vi}^{t}含义同定义 1。

[0068] 定义 4，实际开始时间 AST(v_i)(Actual Start Time)：指任务 v_i在所分配的虚拟机上实际的开始时间。

[0069] 2、虚拟机类型映射

[0070] 在云端有多种类型的虚拟机，每种都有不同的配置，主要区别在于内核数、CPU 频率、存储空间大小、I/O 带宽的不同，表 1 和表 2 分别给出了亚马逊和 Rackspace 公司的云平台虚拟机配置。

[0071] 表 1 亚马逊公司云平台虚拟机主要配置及收费标准

[0072]

虚拟机类型	内核数（个）	内存 (GB)	价格（美元 / 小时）
M1.small	1	1.7	0.06
M1.medium	2	3.75	0.12
M1.large	4	7.5	0.24
M1.xlarge	8	15	0.48

[0073] 表 2 某公司云平台虚拟机配置及收费标准

[0074]

虚拟机类型		配置	价格（美元/小时）
Normal types	Small（N_S）	1.7GB MEM.1 EC2 CU	$0.06
	Medium（N_M）	3.75GB MEM.2 EC2 CU	$0.15
	Large（N_L）	7.5GB MEM.8 EC2 CU	$0.36
	Extra Large（N_EL）	15GB MEM.8 EC2 CU	$0.9
High_memory types	Extra Large（M_EL）	17.1GB MEM 6.5EC2 CU	$0.8
	Double Extra Large（M_DL）	34.2GB MEM 13 EC2 CU	$1.9
	Quadruple Extra	68.4GB MEM 26 EC2 CU	$4.3

[0075]

	Large（M_QEL）		
High-CPU types	Medium（C_M）	1.7GB MEM 5 EC2 CU	$0.145
	Extra Large（C_EL）	7 GB MEM 20 EC2 CU	$0.87

[0076] 由表 1 和表 2 可知，云供应商所提供的虚拟机有多种类型。云供应商所提供的虚拟机按小时进行收费，不同类型的虚拟机收费标准也不相同，不同云供应商收费也不同。表 2 中分 Normal type、High-memory type、High-CPU type 是三种类型的虚拟机，分别适用于普通任务、存储密集型任务、CPU 密集型任务。图 1 中任务的执行时间以及执行成本如表 3 所示。

[0077] 表 3 是图 1 中任务的执行时间以及执行成本

[0078]

任务 / 虚拟机类型	V_1		V_2		V_3		V_4		V_5		V_6		V_7	
	Time (m)	Cost ($)	Time (m)	Cost ($)	Time (m)	Cost ($)	Time (m)	Cost ($)	Time (m)	Cost ($)	Time (m)	Cost ($)	Time (m)	Cost ($)
N_S	138	0.14	86	0.09	91	0.09	208	0.21	61	0.06	135	0.14	171	0.17
N_M	69	0.17	43	0.11	45	0.11	104	0.26	30	0.08	61	0.15	85	0.21
N_L	34	0.20	21	0.13	22	0.14	52	0.31	15	0.09	30	0.18	42	0.25
N_EL	17	0.26	10	0.15	11	0.16	26	0.39	7	0.11	15	0.23	21	0.32
M_EL	21	0.28	13	0.17	14	0.18	32	0.43	9	0.12	18	0.24	26	0.35
M_DEL	10	0.32	6	0.19	7	0.22	16	0.51	4	0.13	9	0.29	13	0.41
M_QEL	5	0.36	3	0.22	4	0.25	8	0.57	2	0.14	4	0.29	6	0.43
C_M	27	0.07	21	0.05	17	0.04	83	0.20	14	0.03	135	0.33	42	0.10
C_EL	6	0.09	5	0.07	5	0.06	20	0.29	3	0.04	32	0.46	10	0.15

[0079] 表 3 给出各个任务在不同类型虚拟机上的执行时间。在双向调度算法实现前，我们首先需要解决虚拟机的映射问题，以保证执行成本最小，同时要满足工作流的 deadline 限制，该映射问题是一个混合整数线性规划问题。约定图 1 中的工作流在规定的 deadline 时刻完成，设 deadline 时长为 120min，采用 CPLEX 实现任务与虚拟机类型之间的映射，过程如下：在 eclipse 中引入 CPLEX.jar 文件，通过调用 AddMinimize 添加优化目标 (cost)，并通过 IRange 来定义约束 (time)。映射结果如表 4 所示。

[0080] 表 4 基于表 3 的任务与虚拟机类型映射

[0081]

任务	v_1	v_2	v_3	v_4	v_5	v_6	v_7
虚拟机类型	C_M	C_M	C_M	C_EL	C_M	N_L	C_EL

[0082] 由表 4 知，CPLEX 算法执行后，v_1 被映射到 C_M 类型虚拟机上，v_2 被映射到 C_M 类型的虚拟机上，v_7 被映射到 C_EL 类型的虚拟机。映射结果在接下来所提到的双向调度算法中将被用到。

[0083] CPLEX 实现了把任务映射到高性价比的虚拟机类型上，但是具体是该类型的哪个虚拟机并没有给定。为合理调度到具体的高性价比虚拟机，本文设计了双向调度算法，该算法包括 backward 和 forward 两个子算法。首先，采用 backward 算法把任务调度到相应的虚拟机的具体时间戳，即把所有任务的“最后通牒”时间段规定好。其次，由于在对应的虚拟机上，该任务的具体时间戳的前面必然有大量的空闲时间，为了进一步利用这些空闲时间，根据任务的最早开始时间和最迟开始时间，采用 forward 算法尽最大可能地接近最早开始时间去调度该任务，节省调度时间和用户开支。

[0084] 图 2 中，设用户规定的 deadline 为 120min，计算每个任务最迟结束时间 $LFT(v_i)$，可知，v_7 为 120min，v_5 为 108min，v_6 为 108min，v_3 为 107min，v_2 为 90min，v_4 为 74min，v_1 为 52min。由表 4 可知，v_7 分配到 CE_L 类型的虚拟机上，该虚拟机执行 v_7 只需要 10min，由于 $LST(v_7)$ 为 110min，可把 CE_L1 时间戳为 [110,120] 分配给 v_7。由表 4 可知，v_5 被分配到 C_M 类型虚拟机，v_5 与其直接后继节点所需类型虚拟机不同，因此需要租用 C_M1 虚拟机，同理，将 [94,108] 时间戳分配给 v_5。继续分配 v_6，同理需要租用 N_L1 虚拟机，将 [78,108] 时间段分配给 v_6。接着对 v_3 进行分配，由表 4 可知，v_3 被映射到了 C_M 类型虚拟机上，由于 v_5 已经租用了 C_M1 虚拟机但是在 [93,110] 时间段不空闲，那么只能重新租用新的 C_M 类型虚拟机 C_M2，把时间段在 [90,107] 分配给 v_3。接着对 v_2 进行分配，由表 4 可知，v_2 被映

射到了 C_M 类型虚拟机上，其最迟开始时间到最迟完成时间为 [64,85]，v_2的唯一直接后继节点 v_3也是采用 C_M 类型虚拟机并且在 [64,85] 空闲，即把 v_2分配在该虚拟机 C_M2，时间段为 [69,90]，可以把最迟开始时间和最迟结束时间都向后延迟 5 分钟，因为原本其最迟开始时间到最迟结束时间为 [64,85]，但是由于其孩子节点（即直接后继节点）v_3与其在同一个虚拟机上执行它们之间的传输时间即可省去，所以最迟开始时间和最迟结束时间都会向后推迟。继续分配 v_4，根据 CPLEX 映射关系，其被映射到 C_EL 类型虚拟机上，最迟开始时间到最迟完成时间为 [54,74]。由于 v_4的两个直接后继节点 v_3和 v_6所使用的虚拟机的类型与 C_EL 不同，即只能查看是否有类型为 CE_L 虚拟机在 [54,74] 上空闲，经计算发现 CE_L1 满足要求，将 CEL_1 在 [54,74] 时间段分配给 v_4。最后分配 v_1，其映射虚拟机类型是 C_M 类型，并且最迟开始时间到最迟完成时间为 [25,52]，由于其所有直接后继节点不都是与它同类型，而且其直接后继节点最早的最迟开始时间是 54min，所属任务是 v_4，所以只能找所有 C_M 类型虚拟机在 [25,52] 是否空闲。由图 1 可知，任务 v_1和 v_4之间传输时间为 2min，因此，要留下 2min 为 v_1和 v_4的传输时间，由于 C_M1 和 C_M2 虚拟机均满足分配条件，通过随机选择，把任务 v_1分配给 C_M2。

[0085] 在本发明的一种具体实施方式中，如图 2 所示，所述步骤 1 可以包括：

[0086] 步骤 S101，计算得到所有任务节点的多个工作区间，根据所述工作区间的最迟结束时间 LFT 将多个所述工作区间从大到小进行排序，从所述任务节点的尾节点开始依次执行步骤 S102；

[0087] 步骤S102，当所述任务节点 v_i的所有直接后继节点所分配的虚拟机的类型相同于所述任务节点所映射的虚拟机的类型，且该虚拟机在所述任务节点 v_i的工作区间内空闲，则将所述任务节点 v_i分配到该虚拟机上，否则，执行步骤 S103；

[0088] 步骤 S103，求出任务节点 v_i与所述任务节点 v_i的直接后继节点的传输时间，比较所述任务节点 v_i的多个直接后继节点的最迟结束时间和所述传输时间之和，得出所述和的最小值所对应的直接后继节点；

[0089] 步骤 S104，当所述和的最小值所对应的直接后继节点所分配的虚拟机的类型与所述任务节点 v_i所映射的虚拟机的类型相同，且该虚拟机在所述任务节点 v_i的工作区间内空闲，则将所述任务节点 v_i分配到该虚拟机上，否则，执行步骤 S105；

[0090] 步骤S105，当所述任务节点 v_i所在类型的已经分配的虚拟机在所述任务节点 v_i的工作区间内空闲，则将所述任务节点 v_i分配到该虚拟机中，否则，执行步骤 S106；

[0091] 步骤 S106，重新租用所述任务节点 v_i所在类型的虚拟机，则将所述任务节点 v_i分配到所述虚拟机中。

[0092] 在该 backward 算法中输入参数为初始工作流（包括所有任务及任务间的前后关系、任务之间的传输时间、任务通过 CPLEX 所映射到的虚拟机类型、工作流 deadline）。

[0093] 输出参数为经过 backward 算法后的工作流（包括所有任务通过 CPLEX 所映射的具体虚拟机的执行时间、任务在具体虚拟机上所分配的时间段）。

[0094] 上述的步骤具体为：

[0095] Step1，从工作流尾节点开始向前计算每个任务的 $LST(v_i)$ 以及 $LFT(v_i)$，按任务最迟完成时间递减排序，依次分配每个任务，被分配以后被标记为 assigned，否则标记为 unassigned；

[0096] Step2，如果任务 v_i所有直接后继节点都分配在同一个虚拟机，并且 v_i所映射到虚拟机的类型与后继节点所分配的虚拟机类型相同，同时，在该虚拟机时间间隔 $[LST(v_i)+TT(v_p, v_i), LFT(v_i)+TT(v_p, v_i)]$ 空闲，则把 v_i分配到该虚拟机上并标记为 assigned，否则跳转到 Step3；

[0097] Step3，计算 v_i所有直接后继节点，求出具有最小 $minLFT(v_c)+TT(v_c, v_i)$ 的任务，如果任务 v_c所租用的虚拟机与任务 v_i所映射的虚拟机是同类型，同时在 $[LST(v_i)+TT(v_i, v_c), LFT(v_i)+TT(v_p, v_i)]$ 空闲，则将 v_i分配到该虚拟机上，并标记为 assigned，否则，跳转到 Step4；

[0098] Step4，在所有同类已分配的虚拟机上寻找 $[LST(v_i), LFT(v_i)]$ 是否存在空闲段，如果有，则把 v_i分配到该时段，并标记为 assigned；否则，跳转到 Step5；

[0099] Step5，由于该时间戳找不到可以分配的虚拟机，因此重新租用一个该类的虚拟机，（即与 v_i所映射的虚拟机同类型），把 v_i分配在时段 $[LST(v_i), LFT(v_i)]$ 内；跳转到 Step1，继续分配 unassigned 的任务，直到所有任务都标记为 assigned；

[0100] Step6，返回经过 backward 算法后的工作流（包括任务通过 CPLEX 所映射的具体虚拟机的执行时间、任务在具体虚拟机上所分配的时间段）。

[0101] 在该种实施方式中，所述工作区间从所述任务节点 v_i的最迟开始时间 LST 至传输到该任务节点 v_i的直接后继节点的完成时间之间的区间。

[0102] 在该种实施方式中，在所述步骤 S102 和步骤 S103 中，当所述直接后继节点为所述任务节点 v_i的唯一直接后继节点，且所述直接后继节点所分配的虚拟机的类型相同于所述任务节点 v_i所映射的虚拟机的类型，则所述工作区间省去相邻节点的传输完成时间。

[0103] 在本发明的一种优选实施方式中，所述步骤 2 可以包括，

[0104] 步骤 S201，将经过 backward 算法之后的工作流中的头节点放入集合 SET 中；

[0105] 步骤 S202，分别计算集合 SET 中的任务节点 v_i在步骤 1 中所分配到的虚拟机以及与所述虚拟机同类的虚拟机的多个空闲时间戳；

[0106] 步骤 S203，根据所述多个空闲时间戳计算得到每个空闲时间所对应的以下因素：最早开始时间影响因素 ESTF、浪费时间影响因素 WTF 和额外租用的费用影响因素 ACF；

[0107] 步骤 S204，通过如下公式得到任务最优放置指数：

[0108] $Index = ESTF \times a + WTF \times b + ACF \times c$，

[0109] 其中，a、b、c 是权重，且 a、b、c 的范围都为 [0,1]；

[0110] 步骤 S205，比较多个所述空闲时间戳所对应的任务最优放置指数的大小得到最小的最优放置指数，将集合 SET 中的任务节点 v_i放入所述最小的最优放置指数所对应的空闲时间戳得到 FeasibleSet 集合；

[0111] 步骤 S206，当所述任务节点 v_i有直接后继节点时，将集合 SET 中的任务节点 v_i替换成所述任务节点 v_i的直接后继节点，否则，执行步骤 S202。

[0112] 基于 backward 算法结果，forward 算法主要从 3 个方面考虑如何选择虚拟机以及时间戳的分配问题，

[0113] (1)ESTF（最早开始时间影响因素）

[0114] 让 AST 与 EST 尽可能地接近，便于后面任务有更多的时间戳去选择，$ESTF = (AST(v_i)-EST(v_i))/LST(v_i)$。在选择时间戳分配时，ESTF 越小越好。

[0115] (2)WTF(浪费时间影响因素)

[0116] 记任务 v_i的开始时间为 ST(v_i),结束时间为 FT(v_i)。租用虚拟机的时间为 HP(分钟),HP = 60*i,i = 1……n。HPleft 代表该虚拟机租用的起始时间,HPright 代表该虚拟机租用的截止时间。当任务 v_i分配到该虚拟机上时,该任务左右各有两种空闲时间戳(分别记为 Slotleft 和 Slotright)。左边的空闲时间戳 Slotleft 的范围是 [Slotstart, ST(v_i)] 或者是 [HPleft,ST(v_i)],同理右边的空闲时间戳的范围是 [FT(v_i),Slotend] 或者是 [FT(v_i),HPright],任务 v_i浪费的总时间戳 Slotsum = Slotleft+Slotright。浪费时间影响因素 WTF 示例如图 3 所示。

[0117] 图 3 中,在 CE_L1 虚拟机上,v_9左边空闲时间戳为 [20,25],而在 CE_L2 虚拟机上中,v_9左边空闲时间戳为 [0,25]。以 60 分钟为一个租赁周期,60 是分水岭,因此,在 CE_L1 虚拟机上中 v2 右边空闲时间戳为 [55,60],而在 CE_L2 虚拟机上中为 [45,50]。

[0118] (3)ACF(额外租用的费用影响因素)

[0119] $$ACF=Addedhour(v_i, slot)*60/(d_{vi}^t+TT(v_p, v_i))$$

[0120] 当任务 v_i在被分配的虚拟机 t 上时,函数 Addedhour(v_i,slot) 用于计算执行任务 v_i所需要增加的额外虚拟机的小时数,d_{vi}^t代表任务 v_i在虚拟机 t 上的执行时间,TT(v_p, v_i) 来表示任务 v_p和任务 v_i之间的传输时间,v_p是 v_i的直接父节点,额外租用的费用影响因素 ACF 的示例如图 4 所示。

[0121] 如果 v_9被分配到图 4 的 CE_L1 虚拟机上,则不需要额外增加收费,因为 v_9利用的是该虚拟机调度 v_8和 v_{10}之间的剩余时间。但是若将 v_9分配到 CE_L2 虚拟机上时,还需要增加一个小时的付费,因为在时间戳为 [60,120] 时,该虚拟机并未分配其它任务,所以仍需租用一个小时去执行 v_9,而在 CE_L1 上即可省去此次费用。

[0122] 定义 5,任务最优放置指数 Index(v_i,slot):上述 ESTF、WTF、ACF 三个因素的结合,将决定任务分配到哪个空闲时间戳,如式 (1) 所示。

[0123] $$Index(v_i, slot) = ESTF*a+WTF*b+ACF*c \quad (1)$$

[0124] 式 (1) 中,a、b、c 是权重,范围为 [0,1],用户根据自己的偏好进行设定,本文 a、b、c 取值均为 1。如果用户对于整个工作流时间要求尽可能快地完成,在满足小于用户 cost 的限制条件下,可以增大权重 a 的值。任务最优放置指数 Index(v_i, slot) 值越小,表示可以提前知道任务浪费的时间戳越小,因此减少因虚拟机时间剩余而浪费的额外费用。

[0125] 定义 6,总费用 $Cost_{sum}$:计算所有已经分配的虚拟机的总费用(向上取整),如式 (2) 所示:

[0126] $$Cost_{sum}=\sum_{t=1}^{n}\left|d_{vi}^t/60\right|\times price(t) \quad (2)$$

[0127] 式 (2) 用于计算租用的 n 个虚拟机的总费用。price(t) 代表第 t 种类型虚拟机每小时的收费标准,d_{vi}^t表示任务 v_i在所分配的 t 类虚拟机上执行所需要的时间(单位:分钟),$\left|d_{vi}^t/60\right|\times price(t)$表示任务节点 v_i在所分配的 t 类虚拟机上执行所需要的总费用。$\left|d_{vi}^t/60\right|$表示 d_{vi}^t 除以 60 并且向上取整,以求出该虚拟机总共使用多少小时,以充分体现了云计算按小时收费特点。

[0128] 定义 7，用户自定义优先级：在满足定义 2 的情况下，根据用户的费用要求以及工作流最后实际完成时间两种因素的优先级，用户有 3 种选择模式：费用优先、工作流最后实际完成时间优先及用户自由选择，如下。

[0129] 1) 费用优先模式

[0130] 该策略调度优先级由高到低的顺序是：先按费用优先调度，其次是按工作流最后实际完成时间进行调度。该策略的调度目标是让调度的费用达到最低，并且最小化工作流最后实际完成时间。该策略首先对所有可行调度方案按照费用进行排序，从中选出所有费用最低的调度方案，在这些方案中按照工作流最后实际完成时间进行排序从中选取时间最小的调度方案。

[0131] 2) 工作流最后实际完成时间优先模式

[0132] 该策略调度优先级由高到低的顺序是：先按工作流最后实际完成时间进行调度，其次是按费用进行调度。该策略的调度目标是让调度的工作流最后实际完成时间达到最小，且费用最小化。该策略首先对所有可行调度方案按照工作流最后实际完成时间进行排序，从中选出所有时间最小的调度方案，在这些方案中按照费用进行排序从中选取费用最小的调度方案。

[0133] 3) 用户自定义模式

[0134] 为满足用户的自身偏好，用户可以根据自己的实际需要，在不违反 deadline 和 cost 限制的同时，选择“最低的成本、更快的执行时间或达到性价比最低”三者之一优先执行，即为用户自定义模式。

[0135] 在该种实施方式中，在所述步骤 S203 中，计算最早开始时间影响因素 ESTF 的方法可以包括：

[0136] 最早开始时间影响因素 ESTF 等于实际开始时间 AST 与最早开始时间之差 LST 除以最迟开始时间 EST。

[0137] 在该种实施方式中，在所述步骤 S203 中，计算浪费时间影响因素 WTF 的方法包括：

[0138] 浪费时间影响因素 WTF 等于在预设周期的时间内左边的空闲时间戳与右边的空闲时间戳的和除以所述预设周期的时间。

[0139] 在该种实施方式中，在所述步骤 S203 中，计算额外租用的费用影响因素 ACF 的方法包括：

[0140] 额外租用的费用影响因素 ACF 等于增加的额外虚拟机的预设周期数乘以预设周期除以所述任务节点 v_i 的工作区间。工作区间包括传输时间加上执行时间。

[0141] 在该种实施方式中，该工作流双向调度方法还包括步骤 3，

[0142] 根据每个虚拟机的费用 price(t) 通过如下公式计算已经分配的虚拟机的总费用 $Cost_{sum}$：

[0143] $$Cost_{sum}=\sum_{t=1}^{n}\left|d_{vi}^{t}/60\right|\times price(t)$$ ，

[0144] 其中，price(t) 代表第 t 种类型虚拟机每小时的收费标准，n 表示有 n 种虚拟机；d_{vi}^{t} 表示任务 v_i 在所分配的 t 类虚拟机上执行所需要的时间且 $d_{vi}^{t}/60$ 向上取整。

[0145] 在该种实施方式中，该工作流双向调度方法还包括：将分配调度后的任务节点 v_i

进行标记。

[0146] 综上所述，以图 2 中工作流为例的 v_5为例，采用以最节省费用为调度目标的 forward 算法调度，如图 5- 图 8 所示，v_5分配之前的虚拟机状态如图 5 所示，图 5 中，v_5放在图 5 的最右边，表示其还没有被分配虚拟机。由于 backward 算法为每个任务设定了最迟开始时间，forward 算法需要将 v_5尽可能早地分配到某虚拟机上。由表 4 可知，v_5被映射到 C_M 类型虚拟机上，又由图 5 可知，所有已经分配的 C_M 类型虚拟机有 C_M1 和 C_M2，这两个虚拟机上总共有 a、b、c 三个空闲时间戳。由 forward 算法可知，由于 c 违反了 v_5的最迟开始时间限制，所以首先排除 c。v_5将在 a 或者 b 中进行分配，排除 c 之后，v_5在虚拟机上有两种可选择状态，如图 6 所示。从图 6 可知，在排除时间戳 c 之后，只能从时间戳 a（见图 5 的 C_M1）或时间戳 b（见图 5 的 C_M2）中选择一个。由于是按时间调度的，因此，可通过计算时间戳 a 和时间戳 b 的综合影响因素指数的“任务最优放置指数”来确定，即计算时间戳 a 和时间戳 b 的任务最优放置指数 Index(v_5, slot)，计算公式见式 (1)，此处，式 (1) 中 a、b、c 取值均为 1。

[0147] 针对时间戳 a，计算 Index(v_5, slot) 过程如下：

[0148] 因为 ESTF = 0、WTF = (31+15)/60 = 0.76，ACF = 60/14 = 4.2，代入式 (1) 得 Index(v_5, slot) = 0+0.76+4.2 = 4.96。

[0149] 针对时间戳 b，计算 Index(v_5, slot) 过程如下：

[0150] 又因为 ESTF = 0，WTF = 19/60 = 0.31，ACF = 0，同理得 Index(v_5, slot) = 0+0.31+0 = 0.31

[0151] 可知，4.96>0.31。由定义 5 可知，任务最优放置指数 Index(v_i,slot) 值越小越好，所以 v_5分配给了图 6 中的 C_M2。v_5被分配后，虚拟机最终状态如图 7 所示。

[0152] 从图 7 可知，v_5由于没有租用 C_M1 虚拟机而节省了费用，否则 v_5将按一个周期（小时）付费，v_5被分配在在虚拟机 C_M2 上，通过和 v_1共同租用 C_M2 的一个周期，从而达到了节省费用的目的，同理，后面的任务继续使用 forward 算法，图 8 给出了虚拟机调度图 2 所示的整个工作流的最终结果。

[0153] 图 8 所示的整个工作流在虚拟机上调度的最终结果是所有调度方案中最节约的一种，在于对每个任务分配虚拟机时都求出综合影响因素指数，通过选出最小的空闲时间戳进行分配，以节省租用虚拟机的费用。

[0154] 在本发明的实施方式中，backward 算法中时间复杂度为 $O(N^2)$ 其中 N 表示任务数，在 forward 算法中，由于出现了空闲时间戳选择，假设总共有 H 个空闲时间戳需要去选择，则双向调度算法的总时间复杂度为 $O(N^2H)$。

[0155] 将 ICPCP 算法、BDA 算法与本文算法进行比较。我们取工作流包含的任务数为 100，共有如表 3 所示的 9 种类型虚拟机。考虑到工作流的复杂度 OS 参数 (OS 是指整个工作流中实际 path 路径条数除以该工作流可以容纳最大的 path 条数)，分别取 OS = 0.2、OS = 0.3、OS = 0.4 三种情况。工作流 deadline 的生成方法：D = Dmin+(Dmax-Dmin)*θ，其中 Dmin 是工作流执行时间的最小值，Dmax 是工作流执行时间的最大值，θ 为系数，取值范围为 0 到 1，以保证取到相对合理的 deadline，实验中取 θ = 0.15、θ = 0.3 两种情况分析。工作流中的任务节点路径，将根据 OS 取值随机产生。

[0156] 当 θ 取 0.15 时，本文算法与其它算法在执行费用方面的对比分析如图 9 所示。

[0157] 从图 9 可以看出，当 θ 取 0.15 时，在 OS = 0.2 和 OS = 0.4 时，本文算法比 ICPCP 算法和 BDA 算法进一步节约了成本，尤其与 ICPCP 算法相比，优势非常明显。

[0158] 随着 θ 增大，本文算法的优势更加明显，如 θ 取 0.3 时，对比图如图 10 所示。

[0159] 由图 10 可知，随着 deadline 限制的逐步减小，本文与 BDA 以及 ICPCP 算法进行比较，平均降低 20％ -60％的成本，体现出来本文方法节约成本的明显优势。

[0160] 由于虚拟机按周期调度，而当任务执行时间小于虚拟机周期时，将会产生剩余时间的虚拟机浪费成本问题。为降低当前云计算中工作流任务调度成本及等待时间，本文提出的双向调度算法，该算法包括了 backward 以及 forward 两部分调度，实现将任务调度到具体类型的虚拟机，该算法充分考虑到云计算环境下的虚拟机按小时收费特点以及任务之间传输时间限制，在调度具体虚拟机时，不仅可满足任务按照用户所规定的 deadline 完成、同时可满足用户支付费用最小的需求。通过实验可以看出，本文算法比 ICPCP 以及 BDA 算法进一步节约了费用，同时本文算法可在不违反 cost 约束的情况下，用户可以根据偏好自行选取相应的调度模式。

[0161] 本方法仿真实验的硬件环境为 :Intel Core 1.86GHz CPU，4G 内寸，320G 硬盘，编程语言为 Java。软件环境为 win7 操作系统，Eclipse 以及云计算仿真软件 Cloudsim 仿真器。Cloudsim 是由澳大利亚墨尔本大学的网络实验室研究发布的，对云计算开发和仿真具有很好的效果。用 ILOG CPLEX 解决任务与虚拟机类型的匹配的最小线性二乘问题。

[0162] 以上结合附图详细描述了本发明的优选实施方式，但是，本发明并不限于上述实施方式中的具体细节，在本发明的技术构思范围内，可以对本发明的技术方案进行多种简单变型，这些简单变型均属于本发明的保护范围。

[0163] 另外需要说明的是，在上述具体实施方式中所描述的各个具体技术特征，在不矛盾的情况下，可以通过任何合适的方式进行组合，为了避免不必要的重复，本发明对各种可能的组合方式不再另行说明。

[0164] 此外，本发明的各种不同的实施方式之间也可以进行任意组合，只要其不违背本发明的思想，其同样应当视为本发明所公开的内容。

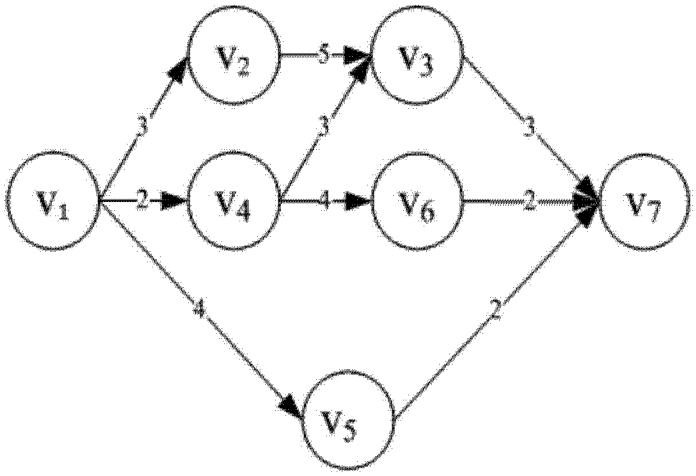

V1
V2
V3
V4
V5
V6
V7
3
5
3
2
3
4
4
2
2

图 1

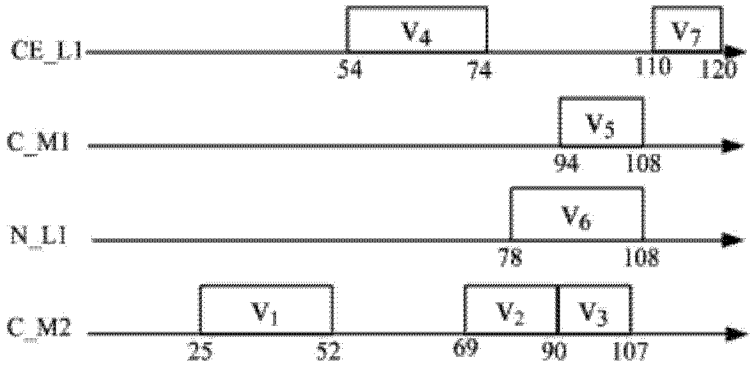

CE_L1
V4
V7
54
74
110
120
C_M1
V5
94
108
N_L1
V6
78
108
C_M2
V1
V2
V3
25
52
69
90
107

图 2

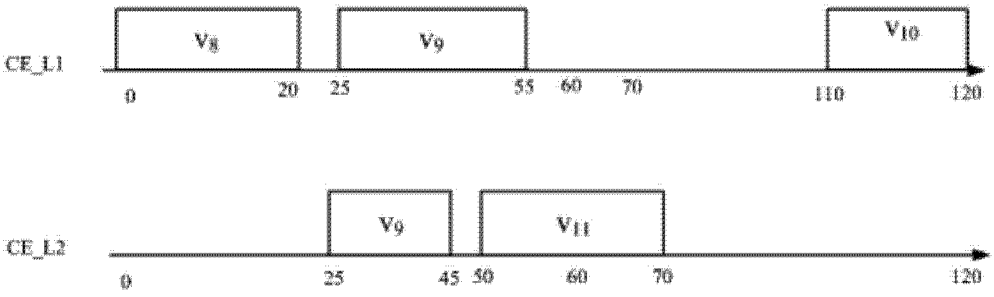

CE_L1
V8
V9
V10
0
20
25
55
60
70
110
120
CE_L2
V9
V11
0
25
45
50
60
70
120

图 3

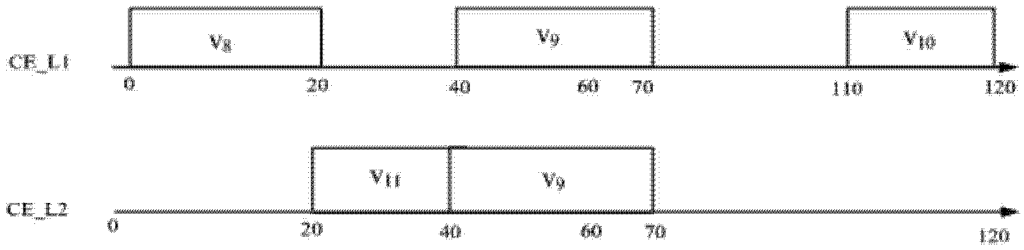

CE_L1
V8
V9
V10
0
20
40
60
70
110
120
CE_L2
V11
V9
0
20
40
60
70
120

图 4

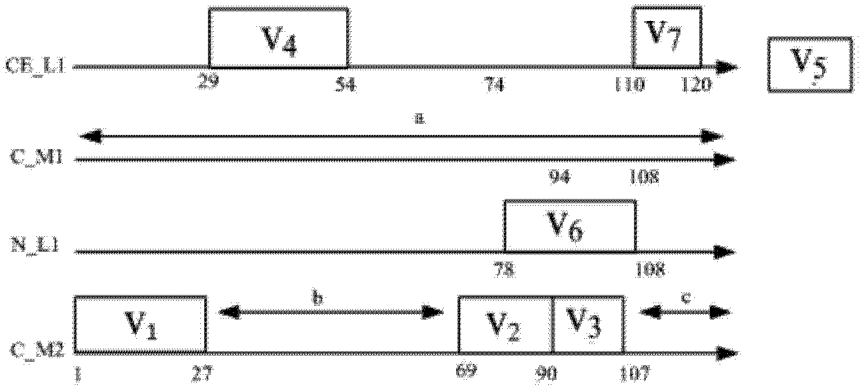

CE_L1
V4
V7
V5
29
54
74
110
120
a
C_M1
94
108
N_L1
V6
78
108
b
c
C_M2
V1
V2
V3
1
27
69
90
107

图 5

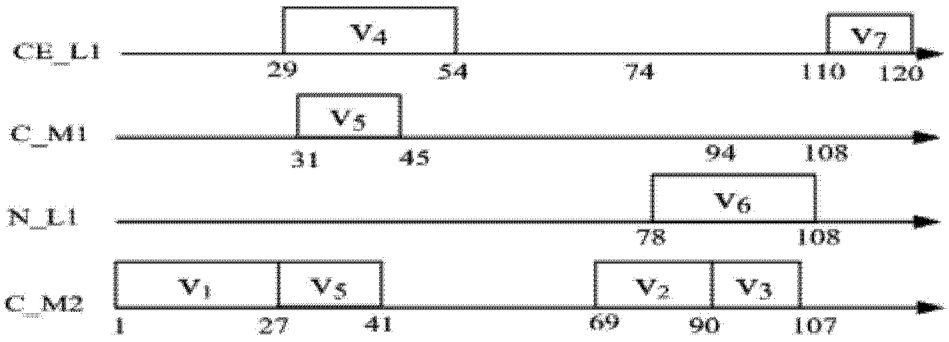

CE_L1
V4
V7
29
54
74
110
120
C_M1
V5
31
45
94
108
N_L1
V6
78
108
C_M2
V1
V5
V2
V3
1
27
41
69
90
107

图 6

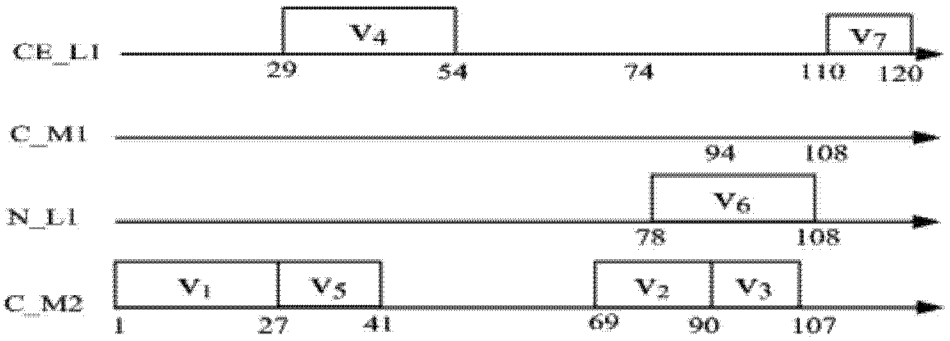

CE_L1
C_M1
N_L1
C_M2
V4
V7
V6
V1
V5
V2
V3
29
54
74
110
120
94
108
78
108
1
27
41
69
90
107

图 7

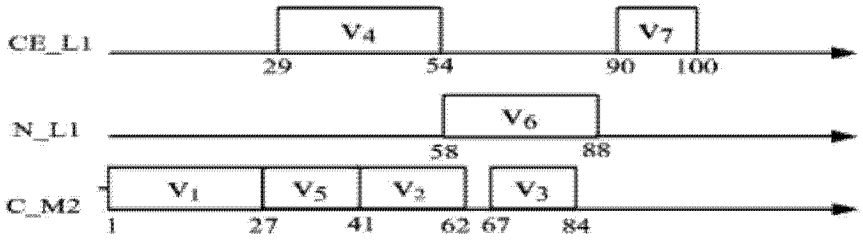

CE_L1
N_L1
C_M2
V4
V7
V6
V1
V5
V2
V3
29
54
90
100
58
88
1
27
41
62
67
84

图 8

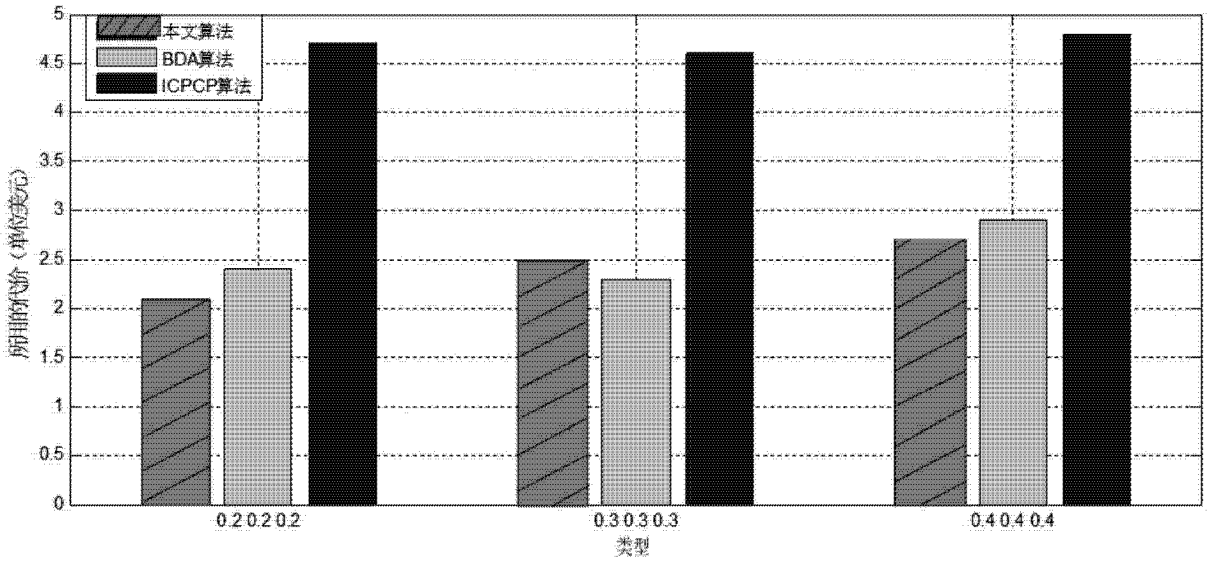

本文算法
BDA算法
ICPCP算法
所用的代价（单位美元）
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0.2 0.2 0.2
0.3 0.3 0.3
0.4 0.4 0.4
类型

图 9

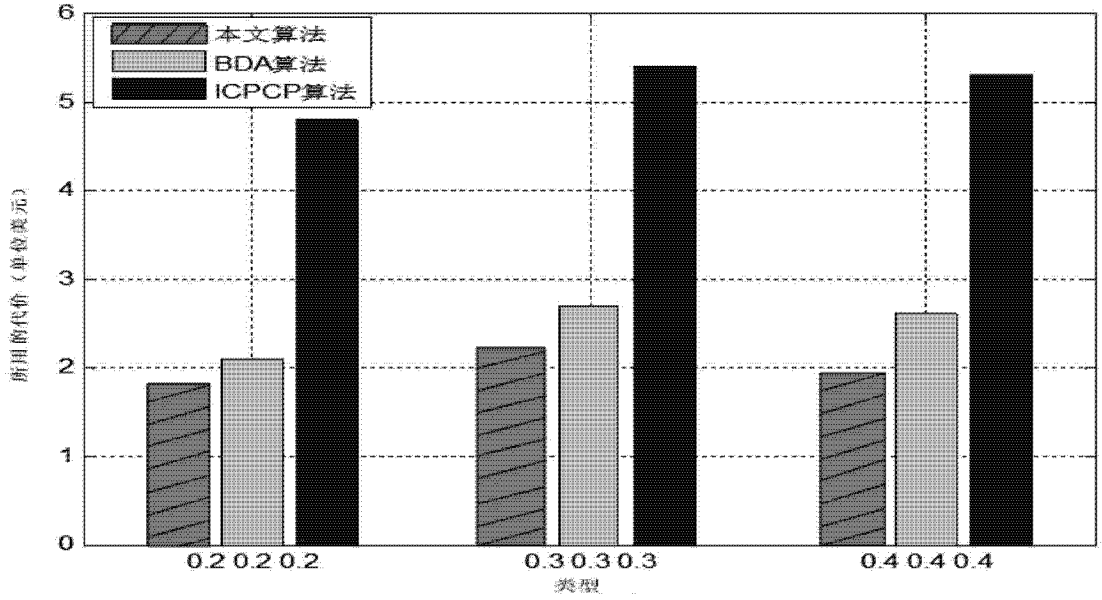

本文算法
BDA算法
ICPCP算法
所用的代价（单位美元）
0
1
2
3
4
5
6
0.2 0.2 0.2
0.3 0.3 0.3
0.4 0.4 0.4
类型

图 10

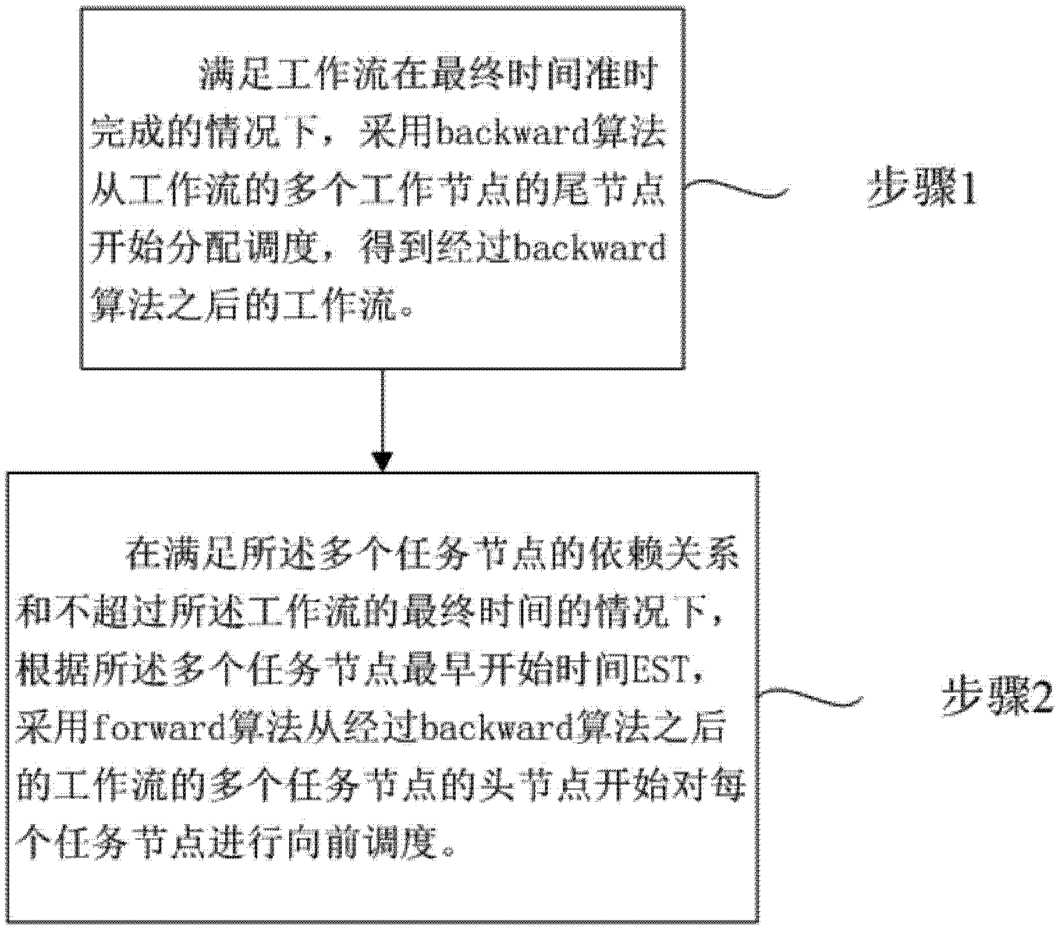

满足工作流在最终时间准时完成的情况下，采用backward算法从工作流的多个工作节点的尾节点开始分配调度，得到经过backward算法之后的工作流。
步骤1
在满足所述多个任务节点的依赖关系和不超过所述工作流的最终时间的情况下，根据所述多个任务节点最早开始时间EST，采用forward算法从经过backward算法之后的工作流的多个任务节点的头节点开始对每个任务节点进行向前调度。
步骤2

图 11